Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Samo Turšič

Trgovalni sistem nad digitalno valuto Bitcoin

diplomsko delo

univerzitetni študijski program prve stopnje
računalništvo in informatika

Mentor: doc. dr. Dejan Lavbič

Ljubljana, 2014

Besedilo je oblikovano z urejevalnikom besedil LaTeX.

Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Tematika naloge:

Pri aktivnem trgovanju na klasičnih borznih trgih si številni uporabniki pomagajo s tehnično analizo, ki temelji na prepričanju, da se zgodovina pogosto ponavlja in če opredelimo vzorce, nam le-ti lahko pomagajo pri napovedovanju gibanja cene (rast, padec, stagnacija) v prihodnje. Obstajajo številne strategije, ki so primerne za določene panoge in načine vlaganja. V zadnjem času je zelo veliko pozornosti namenjeno virtualni denarni valuti Bitcoin. Nekateri verjamejo, da je to denarna valuta prihodnosti, spet drugi pa so mnenja, da gre zgolj za balon, ki bo kmalu počil. To dejstvo potrjuje tudi zelo velika volatilnost njegove vrednosti, ki med potencialne vlagatelja vnaša strah in nezaupanje. V okviru diplomske naloge naj študent pripravi prototip trgovalne platforme, kjer preizkusi različne trgovalne strategije za napovedovanje gibanja vrednosti valute Bitcoin v prihodnosti. Sistem naj omogoča vnos poljubne strategije in testiranje le-te na zgodovinskih ter trenutnih podatkih.

Izjava o avtorstvu diplomskega dela

Spodaj podpisani Samo Turšič, z vpisno številko **63090165**, sem avtor diplomskega dela z naslovom:

Trgovalni sistem nad digitalno valuto Bitcoin

S svojim podpisom zagotavljam, da:

- sem diplomsko delo izdelal samostojno pod mentorstvom doc. dr. Dejana Lavbiča,
- so elektronska oblika diplomskega dela, naslov (slov., angl.), povzetek (slov., angl.) ter ključne besede (slov., angl.) identični s tiskano obliko diplomskega dela,
- soglašam z javno objavo elektronske oblike diplomskega dela na svetovnem spletu preko univerzitetnega spletnega arhiva.

V Ljubljani, dne 5. septembra 2014 Podpis avtorja:

Zahvaljujem se mentorju doc. dr. Dejanu Lavbiču za korekten odnos ter vso pomoč in koristne nasvete pri izdelavi diplomskega dela. Prav tako bi se zahvalil svoji družini, ki me je podpirala ves čas študija.

Kazalo

Seznam uporabljenih kratic

kratica	angleško	slovensko
AJAX	Asynchronous JavaScript and XML	asinhroni JavaScript in XML
API	Application Programming Interface	programski vmesnik
BSON	Binary JSON	binarni JSON zapis
BTC	Bitcoin	bitcoin
CSS	Cascading Style Sheets	prekrivni slogi
CSV	Comma-Separated Values	podatki, ločeni z vejico
EJS	Embedded JavaScript	vključeni JavaScript
EMA	Exponential Moving Average	eksponentno drseče povprečje
HTML	Hyper Text Markup Language	označevalni jezik za izdelavo spletnih strani
HTTP	Hypertext Transfer Protocol	protokol za izmenjavo hiperteksta
IP	Internet Protocol	spletni komunikacijski protokol
Java EE	Java Enterprise Edition	platforma Java za poslovna okolja
Java ME	Java Micro Edition	platforma Java za majhne naprave
Java SE	Java Standard Edition	standardna verzija platforme Java
JSON	JavaScript Object Notation	JavaScript objektna notacija
JVM	Java Virtual Machine	javanski navidezni stroj
MA	Moving average	drseče povprečje
MACD	Moving Average Convergence Divergence	konvergenca in divergenca drsečih povprečij

MVC	Model–View–Controller	model-pogled-kontroler
NoSQL	Not only SQL	nerelacijske podatkovne baze
OHLC	Open-High-Low-Close	najvišja-otvoritvena-najnižja -zaključna (vrednost)
P2P	Peer-to-Peer	omrežje vsak z vsakim
PIN	Personal Identification Number	osebna identifikacijska številka
RSI	Relative Strength Index	indeks relativne moči
SMA	Simple Moving Average	preprosto drseče povprečje
SQL	Structured Query Language	strukturirani povpraševalni jezik
SVG	Scalable Vector Graphics	umerljiva vektorska grafika
USD	United States Dollar	ameriški dolar
VML	Vector Markup Language	opisni jezik za prikaz vektorske grafike

Povzetek

V okviru diplomskega dela je bila izdelana informacijska rešitev, ki omogoča izvajanje različnih trgovalnih strategij nad kriptovaluto Bitcoin. Podprte borze so Bitstamp, Btc-e ter MtGox. Na področju tehnične analize za Bitcoin že obstajajo razne rešitve, ki trgovalcem pomagajo pri trgovanju in jim svetujejo s pomočjo tehničnih indikatorjev in vzorcev. Vendar ima vsaka rešitev svoje slabosti, ki se jih je želelo odpraviti. Razvita je bila spletna aplikacija v tehnologiji Node.js, ki poleg izvajanja strategij na zgodovinskih podatkih za vsako podprto borzo ponuja še prikaz grafa vrednosti kriptovalute skozi čas z japonskimi svečniki ter grafa globine trga skupaj s knjigo naročil. Za vizualizacijo grafov se je uporabilo knjižnico Highstock. Uporabnik ima možnost implementirati tudi lastno strategijo s pomočjo knjižnice Talib, ki vsebuje funkcije za računanje tehničnih indikatorjev. Ob izvedbi strategij se trgovalni signali izrišejo tudi na graf. Trgovalni podatki, ki so potrebni za delovanje spletne aplikacije in morajo biti osveženi, se s podprtih borz pridobivajo preko vmesnikov API s pomočjo javanskih programov in shranjujejo v podatkovno bazo MongoDB.

Ključne besede: Bitcoin, tehnična analiza, tehnični indikator, izvajanje trgovalnih strategij, japonski svečniki, Node.js, MongoDB, pridobivanje podatkov, Java.

Abstract

In this thesis an information solution was developed that enables the implementation of different trading strategies and backtesting over cryptocurrency Bitcoin trading data. Supported exchanges are Bitstamp, BTC-e and Mt-Gox. In the field of technical analysis there already exist various solutions for Bitcoin that help traders to trade and advise them on basis of technical indicators and patterns. However, each has its own drawbacks, which we are aiming to fix. A web application was developed in Node.js technology, which in addition to the backtesting for each supported exchange also displays a chart that shows the value of cryptocurrency over time with Japanese candlesticks and market depth chart along with orderbook. Highcharts charting library was used for all visualization. The user has the option to implement its own strategy with the help of Talib library, which contains functions for calculating technical indicators. When backtesting strategies, trading signals are also plotted on a chart. Trading data, which are necessary for web application to function properly are obtained using APIs of supported exchanges with the help of Java programs and are stored in MongoDB database.

Keywords: Bitcoin, technical analysis, technical indicator, backtesting, Japanese candlestick, Node.js, MongoDB, data extraction, Java.

Poglavje 1

Uvod

V zadnjih letih smo priča velikemu vzponu kriptovalut. Te postajajo prepoznavne v širši javnosti predvsem zaradi hitre rasti vrednosti, špekulativnih balonov in uporabi v spletnem podzemlju, ki jo spodbuja anonimnost poslovanja z omenjenimi valutami [2]. Bitcoin (BTC) je prva in tudi najbolj popularna kriptovaluta, ki je decentralizirana, anonimna in omogoča takojšnje in varne spletne transakcije [1]. Trgovanje z bitcoin kovanci postaja vse bolj priljubljeno in razširjeno. Z vsakim dnem jih kot plačilno sredstvo sprejema vedno več podjetij po celem svetu. Takšna podjetja imamo tudi v Sloveniji - po trenutnih podatkih naj bi jih bilo že vsaj 50 [1]. Ne glede na to, da gre za začetne korake, pa ni mogoče zanikati Bitcoinovega velikega potenciala, da v prihodnosti postane resna alternativa klasičnim valutam.

Z bitcoin kovanci je mogoče trgovati na številnih borzah, ki so namenjene izključno trgovanju s to kriptovaluto. Omenjene borze zaradi dostopnosti in izjemno nizkih provizij posameznikom omogočajo, da hitro in enostavno postanejo borzni trgovci [2]. Uporabniki teh borz pa se morajo zavedati raznih tveganj, ki lahko nastanejo pri trgovanju s to kriptovaluto. Bitcoin trenutno ni priznan kot uradna valuta s strani večine držav, torej uporabnikov zakonodaja ne varuje, pojavljajo pa se tudi številni hekerski vdori. Dodatno tveganje predstavljajo tudi velika valutna nihanja. Zaradi tega se pojavlja veliko špekulantov, ki izkoriščajo velika nihanja vrednosti kriptovalute in raz-

like med vrednostmi na različnih borzah, saj to lahko prinaša velike zaslužke. Kaj hitro pa lahko tudi izgubijo velik del premoženja. Vlagatelji seveda želijo vedeti, kdaj kupiti in kdaj prodati. Tako za uspešno trgovanje potrebujejo tudi rešitve, ki bi jim pomagale pri trgovanju, jim svetovale ter jim tako povečale možnosti za uspeh. Različne rešitve [3] že obstajajo za razna forex trgovanja, trgovanja z delnicami in surovinami. Gre za rešitve, ki ponujajo nasvete na podlagi trgovalnih strategij, ki uporabljajo različne tehnične indikatorje. Ti skušajo napovedati prihodnje ravni vrednosti ali pa preprosto splošni trend na podlagi preteklih vzorcev. Prav zaradi popularnosti in zanimivosti kriptovalute Bitcoin je bila to osrednja motivacija diplomskega dela - razviti podoben sistem za Bitcoin.

Cilj diplomskega dela je razviti prototip informacijske rešitve, ki omogoča izvajanje različnih trgovalnih strategij nad kriptovaluto Bitcoin. Rešitev bomo razvili v tehnologiji Node.js kot spletno aplikacijo. Pri tem bi podprli tri največje borze: Bitstamp, Btc-e in MtGox.[1] Najprej je treba preučiti, kako pridobiti podatke o trgovanju s podprtih borz. Pri tem mislimo tako na zgodovinske kot trenutne trgovalne podatke. Ti podatki se nato uporabijo za graf japonskih svečnikov, ki prikazuje vrednost kriptovalute Bitcoin skozi čas. Ob izvajanju različnih strategij se generirani trgovalni signali vizualizirajo tudi na grafu. Aplikacija poleg že implementiranih strategij omogoča uporabniku, da napiše tudi svojo. Ker nas kot vlagatelja zanima tudi globina trga posamezne borze, bo naša rešitev vključevala tudi prikaz knjige naročil ter vizualizacijo globine trga.

V naslednjem poglavju diplomskega dela bomo pregledali področja, ki so pomembna za razumevanje vsebine diplomskega dela, in raziskali obstoječe sorodne rešitve. Tretje poglavje ponuja opis tehnologij in orodij, ki smo jih uporabili pri razvoju informacijske rešitve. Tukaj bodo opisane tudi dodatne knjižnice. V četrtem poglavju pa bosta predstavljena razvoj in implementacija informacijske rešitve. Povsem na koncu v zadnjem poglavju pa sledijo še sklepne ugotovitve, kjer so opisane tudi mogoče izboljšave.

[1] Borza MtGox je v času pisanja diplomskega dela bankrotirala in prenehala delovati.

Poglavje 2

Pregled področja in sorodnih rešitev

2.1 Kriptovaluta Bitcoin

Beseda kriptovaluta označuje vsako digitalno valuto, ki temelji na kriptografiji in je decentralizirana. Prvi delni poskus razvoja kriptovalute je bil sistem DigiCash, ki je bil ustvarjen leta 1990 za anonimizacijo transakcij. Vendar je bil ta sistem centraliziran in je kmalu tudi propadel. Oznaka kriptovaluta se je prvič začela uporabljati šele s prihodom sistema Bitcoin leta 2009, ki je prva uspešna implementacija koncepta kriptovalute, kot ga je prvi opisal Wei Dai leta 1998 v svojem zapisu na elektronskopoštni listi "Cypherpunks". Od prihoda kriptovalute Bitcoin je potekal hiter razvoj kriptografskih metod za plačevanje. Danes sta poleg kriptovalute Bitcoin, ki jo bomo predstavili v tem poglavju, popularni še kriptovaluti Litecoin ter Namecoin [4, 7].

2.1.1 Splošno

Bitcoin je eksperimentalna in decentralizirana kriptovaluta, ki omogoča takojšnje transakcije komurkoli in kamorkoli po svetu [8]. Projekt Bitcoin je ustvaril Satoshi Nakamoto, ko je podal specifikacije in dokaz koncepta v svojem članku "Bitcoin: A Peer-to-Peer Electronic Cash System" [5]. Po-

membne značilnosti vključujejo anonimnost in decentralizirano P2P omrežno strukturo, ki verificira Bitcoin transakcije na podlagi kriptografije namesto nekega centralnega organa ali banke, kar povečuje varnost [6, 7].

Slika 2.1: P2P omrežje (omrežje vsak z vsakim).

Ena od prednosti kriptovalute Bitcoin je, da se ne zanaša na centralni organ ali banko. Namesto tega celotno omrežje sledi in potrjuje transakcije. Ločuje tudi račun od identitete, tako da so transakcije lahko anonimne. Dva uporabnika lahko tako naredita transakcijo brez poznavanja identitete ali lokacije drug drugega. [6].

Enote kriptovalute Bitcoin se imenujejo bitcoin kovanci (BTC) in jih lahko naprej razdelimo na osem decimalnih mest. Služijo isti funkciji kot običajni denar. Z njimi lahko kupimo izdelke in storitve, kjer jih sprejemajo kot plačilno sredstvo. Lahko jih pošljemo nekomu brez posredovanja plačilnih sistemov, zamenjamo za pravi denar, doniramo, investiramo ali kaj drugega.

Trenutno Bitcoin sistem še ni preveč razširjen, ni veliko trgovin oziroma podjetij, ki bi sprejemala bitcoin kovance. Bolj kot za osnovno uporabo je zaradi svoje volatilnosti in velikih razlik v vrednosti na različnih borzah priljubljen pri špekulantih, ki skušajo to izkoristiti za zaslužek.

Večina borz ima na voljo tudi vmesnik API, ki omogoča pridobivanje jav-

nih podatkov, uporabnikom pa tudi upravljanje njihovega računa. S pomočjo teh API-jev je mogoče ustvariti razne rešitve za pomoč pri trgovanju, kot so trgovalni roboti, to so programi, ki omogočajo avtomatsko trgovanje.

Poznamo več načinov, kako priti do bitcoinov:

- Najbolj pogost način je nakup preko Bitcoin borz. Najbolj znana trenutno je slovenski Bitstamp [9].
- Sprejem kot plačilo za storitev ali blago.
- Sodelovanje pri rudarjenju.

2.1.2 Delovanje

Bitcoin kovanci so shranjeni v Bitcoin denarnicah, ki ustvarjajo naslove, preko katerih lahko dobiš ali pošlješ nakazilo. Novi, različni naslovi se lahko generirajo za različne transakcije, kolikorkrat je potrebno. Ni pametno, da se isti naslov uporabi večkrat [11].

Pošiljanje bitcoinov iz ene denarnice v drugo se izvede z objavo nakazila na omrežju in vključuje izvorni in ponorni naslov ter količino, vse skupaj pa je podpisano s privatnim ključem izvornega naslova. Ta podpis predstavlja matematični dokaz, da je nakazilo res poslal lastnik denarnice. Nakazila kasneje ne more nihče več spreminjati. Informacija o nakazilu se nato razširi po vsem omrežju. Nakazila se združujejo v bloke, ti pa v verigo blokov, ki vsebuje skupek vseh potrjenih nakazil. Odjemalci v omrežju potrjujejo nova nakazila in jih dodajo v verigo blokov. To potrjevanje se imenuje rudarjenje. Cilj je vzdrževanje verige blokov in zaščita omrežja. Rudarjenje pa ni trivialno, ampak predstavlja precej zahteven kriptografski problem, ki ga morajo odjemalci rešiti. V zameno prejmejo nagrado v obliki novih bitcoinov. Zaradi tega je proces tudi dobil ime rudarjenje, saj pri tem nastajajo novi bitcoini. Ko bo dosežena zgornja meja, 21 milijonov bitcoinov, te nagrade ne bo več. [6, 10].

2.1.3 Prednosti in slabosti

Obstaja kar nekaj prednosti kriptovalute Bitcoin, ki jo naredijo posebno in unikatno. Kot vsaka stvar, pa ima tudi Bitcoin nekaj slabosti. V tem podpoglavju bomo predstavili tako prednosti kot slabosti [11].

Prednosti

Preprosta in hitra plačila: Bitcoin plačila se izvajajo preprosto z uporabo aplikacije na telefonu, programske opreme na računalniku ali pa preko spleta. Nobene potrebe ni po kreditnih karticah, PIN kodah ali dokumentih, ki jih je treba podpisovati. Vse, kar je potrebno vedeti, je naslov za nakazovanje sredstev. Nakazila so zelo hitra in so lahko končana v nekaj sekundah. To je možno zaradi decentraliziranega plačilnega sistema, ki ne vključuje bank, ki običajno potrebujejo več dni za izvedbo transferja, še posebno v tujino.

Varnost: Močna kriptografija skrbi za to, da transakcije ostanejo varne. Nihče razen lastnika ne more izvesti nakazil ali plačil iz denarnice.

Anonimnost do neke mere: Identitete oseb, ki stojijo za Bitcoin denarnicami, so anonimne. Denarnice pri transakcijah uporabljajo naslove in tudi ti so kreirani privatno. Pri tem je priporočljivo uporabiti različne naslove za vsako transakcijo. To zagotovi zasebnost in anonimnost vpletenih oseb. Bitcoin transakcije pa so javne, sledljive in za vedno shranjene v Bitcoin omrežju. IP naslovi uporabnikov na Bitcoin omrežju so tudi zabeleženi. Če torej uporabnik uporabi isti naslov za več transakcij, ga je možno izslediti ali pa dobiti njegovo identiteto preko IP naslova. To se lahko prepreči z uporabo različnih naslovov za različne transakcije in skrivanjem svojega IP naslova. Bralce, ki anonimnost sistema Bitcoin zanima bolj podrobno, si lahko preberejo študijo "Analysis of Anonymity in the Bitcoin System" [12].

Nizke provizije ali pa jih sploh ni: Banke ponavadi zaračunajo provizije za opravljene transakcije. Pri kriptovaluti Bitcoin te provizije odpadejo.

Za nakazilo denarja kamorkoli ni potrebno plačati nobenih stroškov ali provizij. V primeru, da ti stroški obstajajo, pa gre vedno za majhne vsote. Pri tem gre ponavadi za prostovoljne dajatve, ki omogočajo hitrejšo potrditev transakcije.

Slabosti

Transakcije so nepreklicne: Pri kriptovaluti Bitcoin ni nobene centralne banke ali agencije, ki bi lahko poskrbela za vrnitev sredstev. Sredstva je po nakazilu mogoče dobiti nazaj le, če jih prejemnik vrne.

Izguba denarnice: V primeru, da oseba izgubi dostop do svoje denarnice, so vsa sredstva v njej izgubljena. Eden od načinov za preprečevanje tega je izdelava kopij in shranjevanje denarnice na več lokacijah.

Eksperimentalna valuta: Bitcoin je eksperimentalna valuta, ki se še vedno razvija. Težko je napovedati njeno prihodnost. Trenutno veliko držav ne obdavčuje Bitcoin transakcij, kar pa se lahko spremeni. Kot plačilno sredstvo je kriptovaluta zelo odvisna od tehnologije, čisto mogoče je, da ima kakšno nedokumentirano tehnično napako, ki bi se jo lahko v prihodnosti zlorabilo. Ena od takšnih napak naj bi se že pojavila, in se imenuje razteglјivost transakcij. Ta naj bi bila neuradno tudi krivec za propad do tedaj največje borze MtGox. Gre za možnost spreminjanja podrobnosti transakcij, ki jih podpis ne zajema, da je videti, kot da se pošiljanje bitcoinov ni zgodilo. Zaradi tega se lahko zgodi, da se bitcoini pošljejo še enkrat in tako prejemnik prejme le te dvakrat [13].

Ilegalna uporaba: Relativna anonimnost omogoča, da se uporablja za nezakonite aktivnosti, kot so kupovanje drog in orožja. Bitcoin je še posebej privlačen kot sredstvo za izogibanje plačevanju davkov, predvsem zaradi prej omenjene anonimnosti in dejstva, da nobena država nima pristojnosti nad njim in zatorej niso predmet obdavčitve.

Velika volatilnost cene: Vrednost kriptovalute Bitcoin niha zelo nepredvidljivo in tudi ni regulirana z nobene strani.

2.2 Tehnična analiza

Metode, uporabljene za analizo finančnih inštrumentov in izvajanje investicijskih odločitev, lahko razdelimo v dve kategoriji: temeljno in tehnično analizo. Temeljna analiza vključuje analizo značilnosti podjetja ali česa drugega in na podlagi tega se določi vrednost. Tehnična analiza [14] pa uporablja drugačen pristop. Pomembni so samo premiki cene na trgu in ne vrednost sama po sebi. Kljub zapletenim orodjem, ki jih pri tem uporablja, tehnična analiza v bistvu proučuje ponudbo in povpraševanje na trgu z namenom ugotovitve trenda oziroma smeri v prihodnosti [15].

Tehnična analiza je torej napovedovanje prihodnjih cenovnih premikov glede na analizo preteklih vrednosti. Namesto ocene oziroma meritve vrednosti finančnih inštrumentov uporablja grafe in razna orodja za identifikacijo vzorcev in trendov, ki napovedujejo prihodnjo aktivnost. Tako kot pri napovedovanju vremena tudi tehnična analiza ne more natančno napovedati prihodnosti. Lahko pa pomaga trgovalcem in vlagateljem predvideti, kaj se bo verjetno zgodilo s cenami. Tehnična analiza je primerna za delnice, valute ali kakršenkoli trgovalni inštrument, kjer na ceno vplivata ponudba in povpraševanje [16].

Obstajajo različni tipi tehničnih trgovalcev. Nekateri za svoje odločitve uporabljajo tehnične indikatorje, drugi se zanašajo na vzorce na grafih. Pri tem je eden najbolj pogosto uporabljanih graf japonskih svečnikov [15].

Področje tehnične analize temelji na treh glavnih predpostavkah [15], ki jih je postavil Charles Dow:

- V ceni so že vsebovane vse informacije.
- Cenovni premiki niso popolnoma naključni, ampak sledijo trendom.

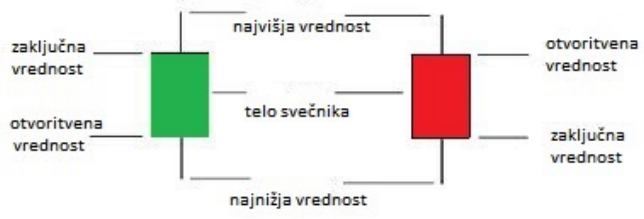
- Zgodovina se nagiba k ponavljanju, predvsem v smislu cenovnih premikov.

Potrebno je pojasniti še dva izraza, ki se pogosto uporabljata v finančnem svetu in tehnični analizi. To sta bik in medved. Bik v splošnem v financah pomeni rast, medved pa padec. Največkrat se uporabljata pri definiranju trenda na trgu, lahko pa tudi za kaj drugega. Medvedji trend tako pomeni, da je trg v padajočem trendu, bikovski pa, da je v rastočem trendu.

2.2.1 Japonski svečniki

Graf japonskih svečnikov [17] je eden izmed najbolj priljubljenih grafov, ki se koristijo za potrebe tehnične analize. Japonski svečniki prikazujejo podatke o najvišji, najnižji, otvoritveni in zaključni vrednosti za določeno časovno obdobje. Ne uporabljajo se samo za prikazovanje cene oziroma cenovnih premikov, ampak tudi kot vizualna pomoč pri trgovanju. Tehnični analitiki jih uporabljajo za določanje, kdaj vstopiti ali izstopiti iz trgovanja. Japonski svečniki namreč tvorijo številne vzorce, na katerih temeljijo trgovalne strategije. Z znanstvenimi testi so bili predstavljeni močni dokazi, da imajo ti vzorci res veliko napovedovalno sposobnost [19].

Telesa svečnikov so dveh barv. V našem primeru, kot je prikazano na sliki 2.2, imajo telo rdeče barve svečniki, katerih zaključna vrednost je manjša od otvoritvene. Takšni svečniki se imenujejo tudi medvedji svečniki. Svečniki s telesom zelene barve oziroma bikovski svečniki pa so tisti, katerih zaključna vrednost je večja od otvoritvene.



Slika 2.2: Japonski svečnik.

Harami vzorec

Poznamo mnogo trgovalnih vzorcev, ki temeljijo na japonskih svečnikih. Eden izmed njih je tudi harami. Harami svečni vzorec [18] je lahko bodisi medvedji bodisi bikovski.

Medvedji harami vzorec nastane, ko večjemu bikovskemu svečniku sledi manjši medvedji ali bikovski svečnik. Najpomembnejši aspekt tega vzorca je cenovni preskok navzdol pri otvoritveni vrednosti drugega svečnika in nezmožnost dviga cene nazaj k zaključni vrednosti prvega svečnika.

Bikovski harami vzorec nastane, ko večjemu medvedjemu svečniku sledi manjši medvedji ali bikovski svečnik. Najpomembnejši aspekt tega vzorca je cenovni preskok navzgor pri otvoritveni vrednosti drugega svečnika in nezmožnost padca cene nazaj k zaključni vrednosti prvega svečnika.

Slika 2.3: Medvedji in bikovski harami vzorec.

Nakupni signal: če se je po pojavitvi bikovskega harami vzorca cena še dodatno zvišala.

Prodajni signal: če se je po pojavitvi medvedjega harami vzorca cena še dodatno znižala.

2.2.2 Tehnični indikatorji

Tehnični indikatorji [20] so ponavadi matematične fomule oziroma algoritmi, ki skušajo napovedati prihodnje ravni vrednosti finančnih inštrumentov ali pa preprosto splošni trend na podlagi preteklih vzorcev in podatkov. Na njih temeljijo različne trgovalne strategije. Poznamo dve vrsti indikatorjev: sledilni in vodilni indikator. Bistvena razlika je v tem, da vodilni pokažejo cenovne premike, preden se ti dejansko zgodijo, sledilni pa šele zatem. V nadaljevanju bomo opisali tehnične indikatorje, ki smo jih uporabili za gradnjo preddefiniranih trgovalnih strategij v naši aplikaciji.

Moving average (MA) - drseče povprečje

Drseče povprečje [21] je široko uporabljan in enostaven indikator, ki sledi trendu v obliki krivulje in pri tem poskuša minimizirati šume, ki nastanejo zaradi nihanja cene. Spada med sledilne indikatorje, saj temelji na preteklih cenah. Dve najbolj pogosto uporabljeni drseči povprečji sta SMA (preprosto drseče povprečje), ki računa povprečje vrednosti v določenem obdobju, in EMA (eksponentno drseče povprečje), ki daje večjo težo nedavnim cenam. Najbolj pogosta uporaba je identifikacija trenutne smeri trenda in iskanje podpornih nivojev. Čeprav je indikator pri trgovanju koristen tudi za samostojno uporabo, pa se ga ponavadi uporablja skupaj z drugimi indikatorji. Prav tako drseča povprečja pogosto tvorijo osnovo ostalih indikatorjev, kot je npr. MACD.

Ker gre za sledilni indikator, drseče povprečje cenovne premike oz. trend pokaže šele, ko se ti zgodijo, torej z zamikom. Večje kot je časovno obdobje za računanje drsečih povprečij, večji je zamik. 200-periodna drseča povprečja imajo večji zamik kot 20-periodna, ker vsebujejo cene za zadnjih 200 period. 20-periodno drseče povprečje je v tem primeru kratkoročno, 200-periodno pa bolj dolgoročno.

Dvigajoče se drseče povprečje pomeni, da je finančni inštrument v pozitivnem trendu, medtem ko padajoče drseče povprečje ponazarja negativen trend. Podobno je navzgor usmerjen trend potrjen z bikovskim prehodom,

ki se zgodi, ko se bolj kratkoročno drseče povprečje dvigne nad bolj dolgoročnega. Navzdol usmerjen trend pa je potrjen z medvedjim prehodom, torej ko kratkoročno drseče povprečje pade pod bolj dolgoročnega.

Preprosto drseče povprečje - SMA je najbolj preprosto drseče povprečje. SMA trenutne časovne periode je aritmetično povprečje zaključnih cen v zadnjih n periodah. SMA za naslednjo periodo spet računamo za zadnjih n period, kar pomeni, da v izračun pride zadnja nova zaključna cena, najstarejša, ki se je računala za prejšnjo periodo, pa odpade. Enako tudi pri ostalih drsečih povprečjih.

Eksponentno drseče povprečje - EMA [23] daje večjo težo nedavnim oziroma novejšim cenam, starejše cene pa imajo čedalje manjšo težo v povprečju. Zaradi tega se lahko hitreje odzove na nihanja cen in spremembe kot SMA. To pa je lahko tudi slabost, saj je tako bolj nagnjeno k lažnim signalom.

Slika 2.4: SMA in EMA.

EMA se računa po naslednji formuli [22]:

$$EMA(trenutni) = ((P(trenutni) – EMA(prejsnji)) * M) + EMA(prejsnji) \tag{2.1}$$

Pri tem P označuje ceno, M pa multiplikator. Ta je odvisen od števila period, za katere računamo EMA, in ga dobimo po naslednji formuli:

$$M = (2/(SteviloCasovnihPeriod + 1)) \tag{2.2}$$

RSI - indeks relativne moči

RSI [23] ali indeks relativne moči je eden izmed najbolj priljubljenih tehničnih indikatorjev, saj je eden redkih vodilnih indikatorjev, kar pomeni, da prepozna spremembo trenda že vnaprej. Gre za indikator, ki meri trenutno moč vrednosti tečaja v primerjavi s preteklimi vrednostmi. Lahko se računa na različno dolga obdobja, največkrat se uporablja 14 period.

Indeks je izračunan po naslednji formuli [24]:

$$RSI = 100 - 100/(1 + RS) \tag{2.3}$$

RS pomeni povprečje zaključnih vrednosti period, ko je cena naraščala (zaključna vrednost nad otvoritveno), deljeno s povprečjem zaključnih vrednosti period, ko je cena padala (zaključna vrednost pod otvoritveno).

Vrednosti indeksa RSI se gibljejo med 0 in 100. Vrednost 0 nam pove, da je vrednost tečaja v zadnjih 14 periodah samo padala (glede na zaključne vrednosti), 100 pa, da je v zadnjih 14 periodah samo naraščala.

Nakupni signal: ko RSI zraste nad črto preprodanosti (nad 30 RSI indeksa).

Prodajni signal: ko RSI pade pod črto prekupljenosti (pod 70 RSI indeksa).

Črti preprodanosti in prekupljenosti sta ponavadi nastavljeni na 30 in 70, vendar se ju lahko spreminja po želji.

Slika 2.5: RSI.

MACD

MACD tehnični indikator [23] kaže spremembe v trendu in spada med bolj priljubljene in učinkovite indikatorje. Ker je zgrajen na drsečih sredinah, je sledilni indikator, kar pomeni, da se spremeni šele po tem, ko se vsaj del neke spremembe v ceni že zgodi. Sestavljen je iz treh glavnih komponent:

- MACD - izračuna se tako, da od 12-periodne hitrejše eksponentne drseče sredine (EMA) odštejemo 26-periodno, ki se počasnejše odziva na spremembe.
- MACD signalna črta - 9 periodna eksponentna drseča sredina (EMA).
- MACD histogram - razlika med MACD in MACD signalno črto.

Parametri 12, 26 in 9 so tipično uporabljeni pri tem indikatorju, vendar se lahko zamenjajo glede na trgovalni stil in cilje.

Nakupni signal: ko se MACD dvigne nad MACD signalno črto.

Prodajni signal: ko MACD pade pod MACD signalno črto.

Slika 2.6: MACD.

2.3 Sorodne rešitve

Kot smo napisali že v uvodu, je bil cilj našega diplomskega dela implementirati trgovalno platformo nad kriptovaluto Bitcoin. Pred začetkom razvoja naše rešitve smo se odločili pregledati, ali mogoče obstaja že kakšna sorodna. Čeprav je Bitcoin razmeroma novo področje, smo našli kar nekaj sorodnih rešitev. Vsaka ima svoje prednosti in slabosti, so pa v določeni meri vplivale tudi na razvoj naše rešitve. V nadaljevanju bomo predstavili dve raznoliki rešitvi, ki sta po funkcionalnostih najbolj podobni naši rešitvi.

2.3.1 Gekko

Gekko [25] je trgovalni robot za kriptovaluto Bitcoin in platforma za izvajanje simulacij na zgodovinskih podatkih (angl. backtesting) na popularnih Bitcoin borzah. Napisan je v programskem jeziku JavaScript in teče na platformi

Node.js. Rešitev Gekko je povsem odprtokodna in jo lahko spreminja vsak. Trenutno je še v fazi razvoja in zato vse funkcionalnosti še ne delujejo, nima pa tudi spletnega vmesnika.

Gekko je trgovalna platforma. Omogoča pravo živo trgovanje s pomočjo trgovalnega robota ali simulacijo trgovanja (brez pravega denarja) na podlagi trgovalnih strategij. Poleg tega tudi shranjuje podatke z borz, ki se lahko uporabijo za simulacije trgovanja na zgodovinskih podatkih z uporabo trgovalnih strategij (angl. backtesting). Ta funkcionalnost zaenkrat še ne deluje in je v fazi razvoja. Gekko že ima implentiranih nekaj strategij, ki temeljijo na tehničnih indikatorjih. Parametri indikatorjev, ki so uporabljeni v strategijah, so konfigurabilni. Poleg tega Gekko tudi ponuja možnost napisati svojo trgovalno strategijo. Pri tem pa je potrebno imeti vsaj nekaj znanja programiranja.

Gekko vsebuje tudi sistem vtičnikov, ki izvajajo določene naloge, ko se kaj zgodi, ali pa mu omogočijo, da komunicira z drugimi platformami. Eden izmed vtičnikov je Mailer, ki avtomatsko pošlje elektronsko pošto, ko ima uporabljena trgovalna strategija nov nasvet v obliki prodajnega ali nakupnega signala.

V primerjavi z našo rešitvijo Gekko nima spletnega vmesnika in teče v ukazni vrstici. Prav tako zaenkrat ne deluje simulacija trgovanja na zgodovinskih podatkih. To rešitev smo sprva nameravali uporabiti kot pomoč pri pridobivanju podatkov z borz, vendar smo ugotovili, da rešitev ne deluje zadovoljivo.

2.3.2 Cryptotrader

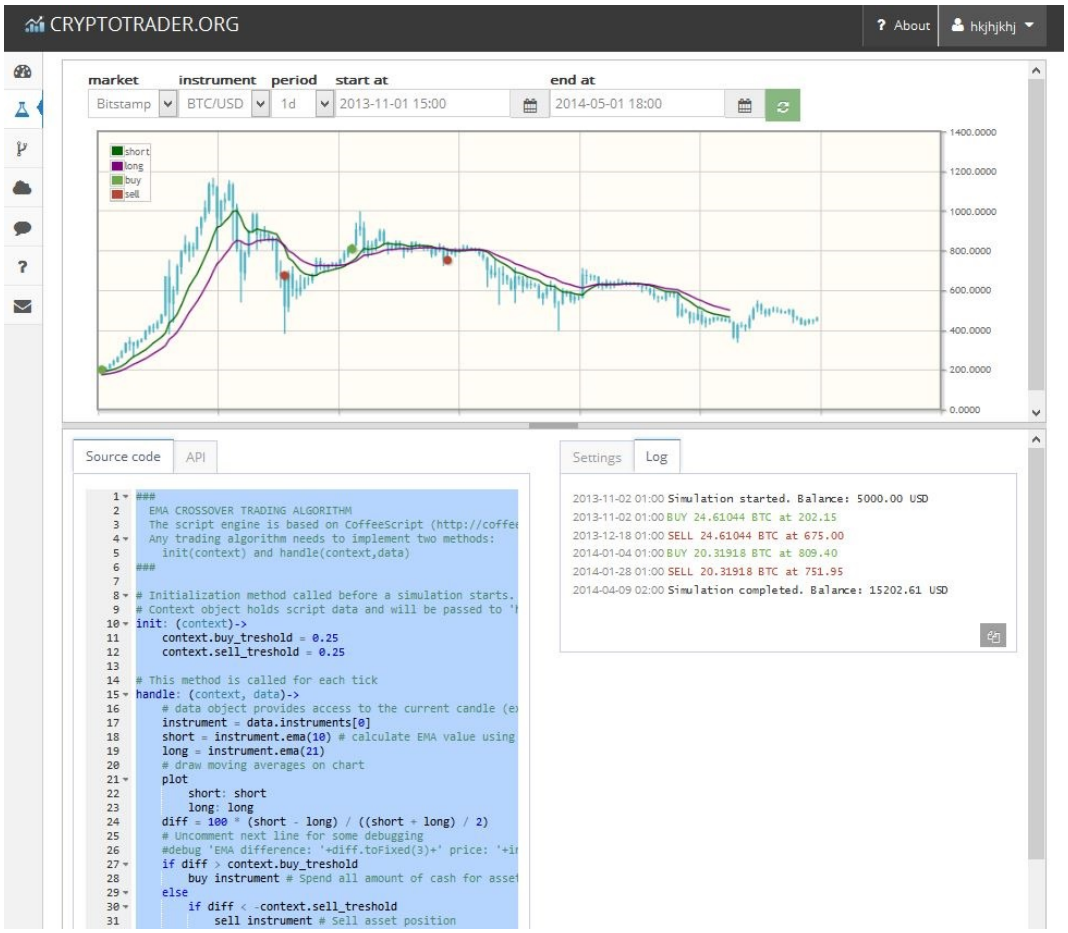
Cryptotrader [26] je celovita spletna trgovalna platforma za Bitcoin in ostale kriptovalute.

Glavne funkcionalnosti:

- Avtomatizirani trgovalni roboti v oblaku - ni potrebe po nalaganju programske opreme, roboti tečejo na njihovih strežnikih.

- Tržnica strategij - prostor, kjer se lahko ustvari, kupi in proda strategije.
- Testiranje trgovalnih strategij na zgodovinskih podatkih (angl. backtesting).
- E-poštna opozorila in objave.

Poleg teh funkcionalnosti ponuja tudi dokumentacijo ter mesto za diskusijo. Na sliki 2.7 je prikazan primer simulacije izvajanja strategije na zgodovinskih podatkih. To je tudi ena izmed glavnih funkcionalnosti, ki smo jo implementirali v našo rešitev.



Slika 2.7: Primer simulacije izvajanja strategije na zgodovinskih podatkih.

Cryptotrader je profesionalna platforma, ki omogoča veliko število funkcionalnosti, za razliko od rešitve Gekko je preprostejša za uporabo, njen uporabnik pa ne potrebuje znanja programiranja. Po drugi strani pa gre za zaprto platformo, ki ne omogoča brezplačne uporabe. Tako kot naša rešitev tudi ta za implementacijo strategij uporablja knjižnico Talib, ki vsebuje funkcije za računanje tehničnih indikatorjev in vzorcev na podlagi japonskih svečnikov.

Poglavje 3

Uporabljene tehnologije in orodja

V tem poglavju bomo predstavili tehnologije, programe, orodja in koncepte, ki smo jih uporabili pri razvoju naše informacijske rešitve. Izbrali smo tiste, ki so se nam zdeli najbolj primerni in so med seboj kompatibilni.

3.1 Programski jeziki in tehnologije

3.1.1 Java

Java [27] je moderni računalniški programski jezik, ki je objektno usmerjen, na razredih temelječ in prenosljiv. Razvit je bil s strani podjetja Sun Microsystems leta 1995. Močno spominja na programska jezika C in C++, saj je njegova sintaksa v večji meri izpeljana predvsem iz teh dveh jezikov. Njihov moto pri razvoju je bil "write once, run anywhere" (napiši enkrat, poženi kjerkoli), s katerim so hoteli pokazati prednosti prenosljivosti med platformami. Javanske aplikacije tečejo neodvisno od računalniške arhitekture ali operacijskega sistema na Java navideznem stroju (angl. Java virtual machine - JVM). S tem so zagotovili prenosljivost, stranski učinek tega pa je malce počasnejše delovanje. Kljub podobnemu imenu in sintaksi Java ni povezana s programskim jezikom JavaScript. Gre za dva neodvisna projekta.

Glavni cilji [28] pri razvoju Jave:

1. Objektna usmerjenost in preprostost.
2. Robustnost in varnost.
3. Neodvisnost od platforme in prenosljivost.
4. Dinamičnost in nitnost.
5. Visoka zmogljivost.

Danes je eden izmed najbolj razširjenih programskih jezikov. Najdemo jo praktično povsod, od računalnikov do podatkovnih centrov, mobitelov, pametnih naprav, interneta. Zaradi različnih ciljnih platform je nastalo več verzij:

- Java ME (Java Platform, Micro Edition) - namenjena majhnim napravam (mobiteli, vgrajene naprave,...).
- Java SE (Java Platform, Standard Edition) - standardna verzija za delovne postaje.
- Java EE (Java Platform, Enterprise Edition) - poslovna verzija, namenjena poslovnim okoljem.

Različico Java SE bomo uporabili za gradnjo programov, ki bodo zadolženi za pridobivanje in shranjevanje podatkov s podprtih borz.

3.1.2 HTML

HTML (Hyper Text Markup Language) je označevalni jezik, namenjen gradnji spletnih strani. Služi za opisovanje strukture in vsebine spletnih strani z uporabo značk in atributov [29]. Na spletu najdemo tudi validacijske storitve, ki nam preverijo sintaktično pravilnost HTML dokumentov. Ta je pomembna za pravilno delovanje v različnih brskalnikih. Po dolgem razvoju je pred kratkim izšla težko pričakovana različica HTML 5, ki v primerjavi s predhodnikom prinaša kar nekaj naprednih novosti [29]:

- semantično strukturiranje besedila,
- zmogljivejši elementi in oblikovanje,
- integracija multimedije,
- animacija na slikarski površini (ang. canvas).

3.1.3 CSS

CSS (Cascading Style Sheets) so prekrivni slogi, ki služijo kot način kontrole predstavitve HTML dokumentov in so definirane v obliki preprostega slogovnega jezika. Z njimi določimo HTML elementom stil, s katerim spletnemu brskalniku povemo, kako naj te elemente prikaže, in tako poskrbimo za želeni videz in obliko spletnih strani. Sintaksa je ločena od HTML, vendar je lahko vsebovana tudi v HTML dokumentih. Poznamo tri nivoje definiranja slogovnih pol [29]:

- vrstične definicije - določajo stil za posamezni element,
- na nivoju dokumenta,
- zunanje prekrivne sloge - z uporabo na množici dokumentov.

Kadar je uporabljenih več nivojev definiranja sloga, se uporabi bolj specifična. Najbolj specifična je vrstična definicija, ki ima tako prednost pred prekrivnim slogom na nivoju dokumenta, ta pa ima prednost pred zunanjimi prekrivnimi slogi. Glavni namen prekrivnih slogov je ločiti delo razvijalcev in oblikovalcev [29].

3.1.4 JavaScript

JavaScript je skriptni jezik, razvit s strani podjetja Netscape z namenom omogočiti poživitev spletnih strani z interaktivnostjo in dinamičnim izvajanjem. To naredi z manipuliranjem in sodelovanjem s HTML elementi [30]. Koda je interpretirana na strani odjemalca, zato mora imeti uporabnik za

delovanje v brskalniku omogočen JavaScript. Trenutno je najbolj popularen programski jezik na svetu.

Najbolj znana knjižnica v jeziku JavaScript je JQuery, ki omogoča poenostavitev pogostih opravil, kot so efekti, animacije, interaktivni vmesniki, AJAX klici, ki omogočajo asinhrono komunikacijo v ozadju, brez potrebe po ponovnem nalaganju strani [29].

3.1.5 Node.js

Node.js je platforma, zgrajena na Googlovem V8 JavaScript pogonu za enostaven razvoj hitrih in skalabilnih omrežnih aplikacij. Uporablja neblokirajoč, z dogodki voden I/O model, kar jo naredi lahko in učinkovito za aplikacije, ki se izvršujejo v realnem času in uporabljajo veliko podatkov (ang. big data) [31].

Gre za precej novo tehnologijo, ki je v zadnjem času v velikem porastu in uživa veliko podporo s strani razvijalcev. Razvil jo je Ryan Dahl leta 2009, ko je izšla prva verzija. Podprta je na vseh priljubljenih operacijskih sistemih: Mac OS X, Windows in Linux [32]. Ker gre za novo tehnologijo, so spremembe hitre in pogoste. To pa lahko povzroča težave, kot so nestabilnost, težave s kompatibilnostjo med verzijami in razne razvojne težave.

Node.js je privlačen zaradi več razlogov [32]:

- Uporaba JavaScripta tudi na strežniškem delu - Razvijalcem je JavaScript dobro poznan, pisanje kode v istem jeziku na strežniškem in odjemalskem delu pa lahko pomeni hitrejši razvoj in boljšo aplikacijo.
- Preprostost uporabe.
- Enostavna skalabilnost.
- Hitro delovanje - Eden od razlogov za to je Googlov pogon V8, ki je prisoten tudi v Googlovem spletnem brskalniku Chrome. Najpomembnejši razlog pa se skriva v asinhronosti. Funkcije so načeloma vedno

asinhrone in tako ni potrebno čakati na rezultate, ampak se lahko vmes izvaja kaj drugega. Vendar po drugi strani asinhrona in dogodkovno vodena koda povečuje kompleksnost in zapletenost kode. Velik problem je npr. dolgo gnezdenje asinhronih funkcij, kar sem občutil tudi sam.

Na spodnjem primeru je prikazana primerjava med sinhronim in asinhronim pristopom. Pri sinhronem pristopu si rezultati ukazov sledijo po vrsti, program počaka, da se poizvedba konča, in najprej izpiše rezultate, šele nato "Pozdravljen svet!". Pri asinhronem pristopu pa namesto, da bi program čakal na rezultate poizvedbe, nadaljuje z izvajanjem. Zato bo v tem primeru najprej izpisal "Pozdravljen svet!", ko bo na neki točki končal s poizvedbo v bazi, pa bo vrnil rezultat preko "callback" funkcije, v tem primeru *function(rows)*.

```
//sinhron primer
var result = database.query("SELECT * FROM hugetable");
console.log(result);
console.log("Pozdravljen svet");

//izhod:
//1. Izpis rezultatov
//2. Pozdravljen svet!

//asinhron primer
database.query("SELECT * FROM hugetable", function(rows) {
    var result=rows;
    console.log(result);
});
console.log("Pozdravljen svet!");

//izhod:
//1. Pozdravljen svet!
//2. Izpis rezultatov
```

Sinhrone funkcije naj bi uporabljali čim manj, saj gre za slabo prakso, izgubijo pa se tudi vse prednosti platforme Node.js. Tistim, ki prehajajo na

to platformo, je ponavadi vsaj na začetku težko razumeti tak način programiranja.

Node.js zelo dobro deluje s podatkovno bazo MongoDB predvsem zaradi dejstva, da obe uporabljata Javascript in JSON. To kombinacijo bomo uporabili tudi za potrebe naše spletne aplikacije.

3.1.6 Ogrodje Sails.js

Aplikacijo lahko začnemo razvijati popolnoma od začetka, vendar to danes ni pogosta praksa, saj je takšen pristop časovno bolj potraten. Lahko pa uporabimo že pripravljeno ogrodje (angl. framework). Ogrodja že vsebujejo osnovne komponente in funkcionalnosti, ki služijo kot temelj aplikacije in s tem omogočajo hitrejši, enostavnejši ter bolj pregleden nadaljnji razvoj. Eno od takšnih ogrodij za platformo Node.js je tudi Sails.js, ki ga bomo uporabili za gradnjo naše aplikacije.

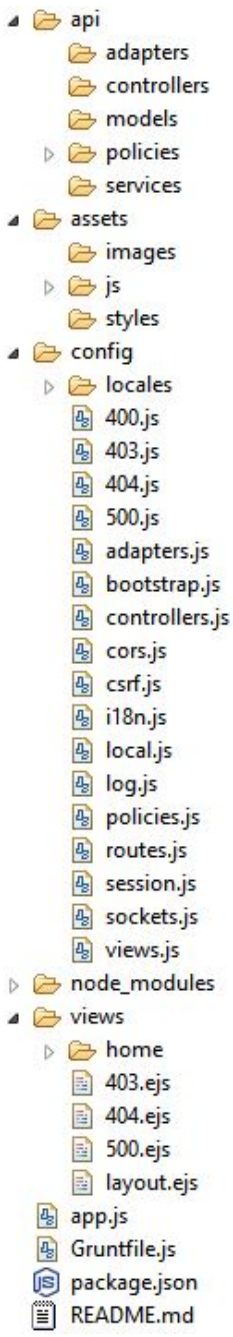
Sails.js [33] je odprtokodno ogrodje za enostavno in hitro gradnjo spletnih aplikacij za platformo Node.js. Ogrodje Sails.js spominja na ogrodje Ruby on rails, saj je bilo zasnovano na njegovi osnovi in prav tako temelji na arhitekturi MVC.

Struktura ogrodja

Ko ustvarimo nov Sails.js projekt, nam ta zgenerira strukturo, prikazano na sliki 3.1.

Mapa api: Ta mapa je sestavljena iz različnih podmap:

- adapters - vključuje adapterje, ki jih uporablja aplikacija za povezave s podatkovnimi bazami,
- controllers - vsebuje kontrolerje aplikacije,
- policies - shranjena pravila za uporabniški dostop,
- services - shranjene api storitve, implementirane s strani aplikacije.



Slika 3.1: Struktura ogrodja Sails.js.

Mapa assets: Sem spadajo vsi statični viri, ki jih uporablja aplikacija, kot so JavaScript, CSS, slike itd. To je tudi najboljši prostor za shranjevanje pomožnih knjižnic, ki jih uporablja aplikacija.

Mapa config: To je ena izmed zelo pomembnih map. Sails.js je načrtovano kot fleksibilno ogrodje. Temelji na določenih standardih, vendar tudi dovoli razvijalcu spremeniti določene nastavitve tako, da bodo ustrezale potrebam projekta. Pomembne datoteke:

- locales - mapa s prevodi,
- adapters.js - konfiguracija adapterjev za podatkovno bazo,
- bootstrap.js - koda, ki se bo izvedla pred zagonom aplikacije,
- controllers.js -nastavitve, povezane s kontrolerji aplikacije,
- local.js - nastavitve, namenjene za lokalni razvoj,
- log.js - konfiguracija nivoja beleženja (informacije, napake itd.) za aplikacijo,
- policies.js - konfiguracija uporabniških pravic,
- routes.js - definiranje poti za aplikacijo,
- session.js - nastavitve, povezane s sejami,
- sockets.js - nastavitve, povezane z vtičnicami,
- views.js - nastavitve, povezane s pogledi aplikacije.

Mapa node_modules: Vsebuje vse node module, ki jih uporabljamo za razvoj aplikacije.

Mapa view: Pogledi aplikacije so shranjeni v tej mapi. Shranjeni so kot EJS (embedded JavaScript) dokumenti. Mapa view vsebuje tudi poglede za lovljenje napak (403, 404, 500) in layout datoteko.

3.1.7 Podatkovna baza MongoDB

Za našo informacijsko rešitev bomo potrebovali tudi ustrezno podatkovno bazo, kamor bomo shranjevali podatke. Ker bodo podatki, ki jih bomo pridobivali, v JSON obliki, se nam je zdelo najbolj primerno izbrati MongoDB, ki namesto tradicionalne relacijske strukture v obliki tabel ponuja hranjenje podatkov v obliki JSON dokumentov (BSON format). Vsi dokumenti so shranjeni v zbirkah (angl. collection). Na sliki 3.2 je prikazan primer dokumenta v MongoDB zbirki ohlcminute, ki vsebuje OHLC podatke, grupirane na minuto.

Slika 3.2: Primer dokumenta v MongoDB.

MongoDB je dokumentno orientirana podatkovna baza, ki omogoča visoko zmogljivost, visoko razpoložljivost in enostavno razširljivost. Je odprtokodna in prenosljiva med vsemi operacijskimi sistemi. Omogoča bogate in dokumentno orientirane poizvedbe. Slika 3.3 prikazuje primerjavo poizvedbe v podatkovnih bazah SQL in MongoDB, ki vrne vse dokumente iz zbirke ohlcminute, kjer je vrednost polja borza (ang. exchange) enaka *bitstamp*. Trenutno je MongoDB najbolj priljubljena in razširjena NoSQL podatkovna baza [34].

```
SELECT *                          db.ohlcminute.find(
FROM ohlcminute                       { exchange: "bitstamp"}
WHERE exchange = "bitstamp"       )
```

Slika 3.3: Primerjava poizvedbe v SQL (levo) in MongoDB (desno).

NoSQL baze [35] za poizvedovanje in shranjevanje podatkov uporabljajo kakšen drug model kot tabelarne relacije pri relacijskih podatkovnih bazah. Zgrajene so tako, da ne potrebujejo vnaprej določene enotne sheme podatkov kot SQL baze, zaradi česar so veliko bolj prilagodljive in omogočajo hitrejše in lažje spremembe. Zaradi visoke zmogljivosti in skalabilnosti v primerjavi z relacijskimi bazami so predvsem primerne za gradnjo aplikacij z veliko količino podatkov (angl. big data) in aplikacij, ki se izvajajo v realnem času.

3.2 Programi

3.2.1 Razvojno okolje Eclipse

Eclipse [36] je odprtokodno razvojno okolje (angl. integrated development environment), ki se uporablja za razvoj aplikacij. Je eno izmed najbolj priljubljenih razvojnih okolij. Namenjeno je predvsem razvoju javanskih aplikacij, vendar lahko s pomočjo različnih dodatkov razvijamo tudi v večini ostalih jezikov, kot so JavaScript, C++, Node.js, Python, Ruby itd. V primerjavi z navadnimi urejevalniki besedil, kot je beležnica, nam ponuja različne uporabne funkcionalnosti, kot so obarvanje kode glede na programski jezik, podporo za razhroščevanje, avtomatsko dopolnjevanje kode, označevanje sintaktičnih napak, drevesni pogled strukture aplikacije, možnost zagona programa kar v razvojnem okolju itd.

To razvojno okolje (verzija Kepler) smo uporabili tako za razvoj aplikacij za pridobivanje podatkov v Javi kot za gradnjo spletne aplikacije v tehnologiji Node.js.

3.2.2 Robomongo

MongoDB ne vključuje grafičnega vmesnika za administracijo in prikaz podatkov, vse se dogaja v terminalu, kot je mongo lupina. Zato smo se odločili, da bomo uporabili še eno orodje, s katerim bomo lažje dostopali do naše baze in nad njo izvajali razne akcije. Izbrali smo Robomongo [37], odprtokodno MongoDB upravljalsko orodje. Vse, kar je mogoče napisati v mongo lupini, je mogoče tudi v Robomongu.

3.3 Dodatne knjižnice

3.3.1 Highcharts JS - Highstock

Highcharts JS [38] je knjižnica za prikaz grafov, napisana v JavaScriptu, ki temelji na SVG in VML prikazovanju. Dokaj hitro je postala ena izmed vodilnih na področju s standardi združljivih JavaScript knjižnic za prikaz grafov. Knjižnica je razdeljena na tri produkte glede na namen vizualizacije: Highcharts, Highstock in Highslide JS.

V naši aplikaciji bomo uporabili knjižnico Highstock, saj gre za produkt, usmerjen v finančno vizualizacijo. Knjižnica je kompatibilna z vsemi sodobnimi brskalniki in ne potrebuje nobenih dodatnih vtičnikov, saj je v celoti napisana v JavaScriptu. Uporabili jo bomo za prikaz grafa vrednosti kriptovalute Bitcoin skozi čas v obliki svečnikov in grafa globine trga.

3.3.2 Talib

Knjižnica Talib [39] je namenjena tehnični analizi finančnih podatkov o trgovanju. Z njeno pomočjo bomo lažje implementirali različne strategije. Vsebuje več kot 100 tehničnih indikatorjev, kot so SMA, MACD, RSI, in omogoča prepoznavo vzorcev na podlagi japonskih svečnikov.

Poglavje 4

Razvoj in implementacija informacijske rešitve

Razvoj smo pričeli z raziskovanjem sorodnih rešitev, ki so nam služile kot opora in motivacija. Nadaljevali smo s preiskovanjem, kje in kako bi pridobili podatke, ki smo jih potrebovali za razvoj našega informacijskega sistema, ter kako bo videti naša podatkovna baza. Zatem smo se lotili programiranja javanskih aplikacij, namenjenih pridobivanju podatkov o trgovanju z različnih borz. Na koncu smo se lotili še programiranja spletne aplikacije v tehnologiji Node.js, pri tem pa smo uporabili ogrodje Sails.js. Nastala informacijska rešitev je spletna platforma za vizualizacijo grafov in simulacijo trgovanja na zgodovinskih podatkih (angl. backtesting) nad kriptovaluto Bitcoin. Trenutno podprte borze so Bitstamp, Btc-e in MtGox. Rešitev se nahaja na spletnem naslovu `sandbox.lavbic.net/bitcoinCharts/`.

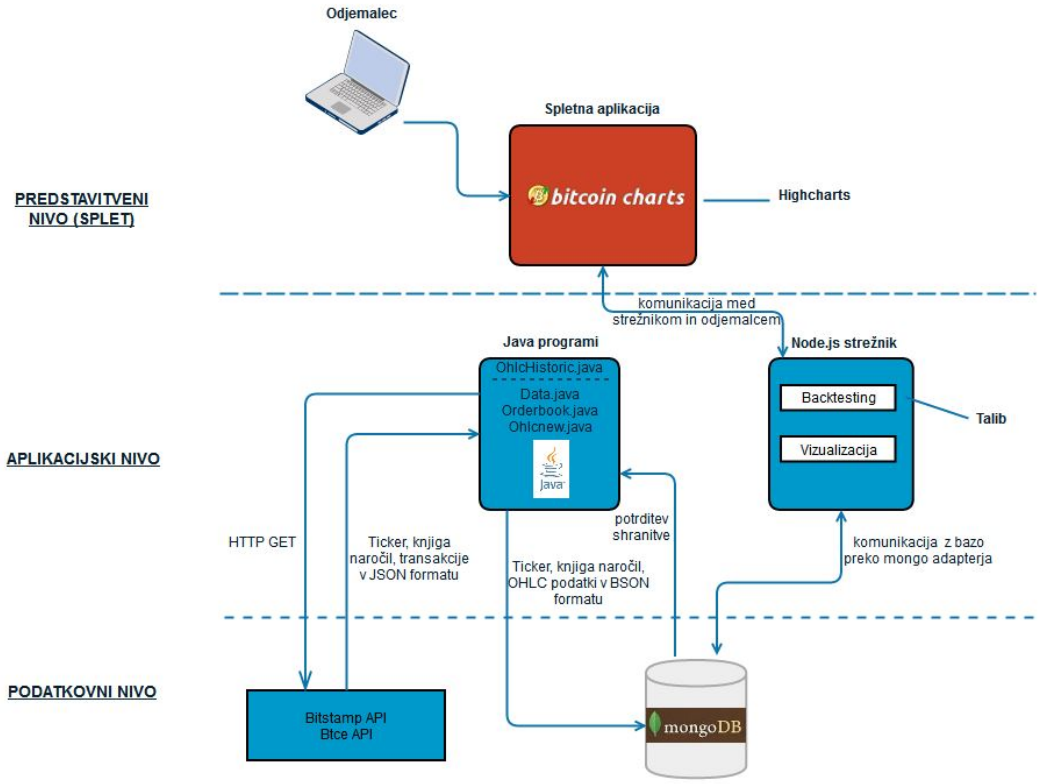
Našo informacijsko rešitev lahko v grobem razdelimo na dva dela:

- pridobivanje podatkov z borz,
- spletna aplikacija.

Prvi del predstavljajo programi v Javi, ki pridobivajo potrebne podatke

z borz, jih obdelajo ter shranjujejo v podatkovno bazo. Te podatke nato uporabi spletna aplikacija za vizualizacijo in izvajanje strategij.

Na spodnji sliki je predstavljena zgradba celotne informacijske rešitve v trinivojski arhitekturi.



Slika 4.1: Zgradba informacijske rešitve.

4.1 Pridobivanje podatkov z borz

Preden smo začeli z izdelovanjem spletne aplikacije, je bilo potrebno pridobiti potrebne podatke z borz ter jih shranjevati v bazo. Najprej smo preverili, na katerih borzah je mogoče trgovati med BTC in USD ter hkrati omogočajo dostop preko vmesnika API. Tako smo se odločili, da podpremo borze Bitstamp, MtGox in Btc-e.

Potrebovali smo naslednje podatke za vsako borzo:

- zgodovinski trgovalni podatki,
- transakcije,
- knjiga naročil,
- ticker.

Vsi podatki razen zgodovinskih so na voljo preko spletnega API-ja borz in jih moramo pridobivati ves čas, saj želimo vedno imeti najnovejše podatke.

Borza MtGox je v času pisanja diplomskega dela bankrotirala in prenehala delovati, zato smo za to borzo pridobili samo zgodovinske podatke.

Izvorna koda vseh programov za pridobivanje podatkov je na voljo na spletnem naslovu `https://bitbucket.org/stursic/savedata`.

4.1.1 Zgodovinski trgovalni podatki

Te podatke smo dobili na spletnem naslovu `http://api.bitcoincharts.com/v1/csv/`, kjer so zbrani zgodovinski podatki večine bitcoin borz v csv datotekah.

```
timestamp, price, amount
1315922016,5.800000000000,1.000000000000
1315922024,5.830000000000,3.000000000000
1315922029,5.900000000000,1.000000000000
1315922034,6.000000000000,20.000000000000
1315924373,5.950000000000,12.452100000000
1315924504,5.880000000000,7.458000000000
1315924614,5.880000000000,0.176882380000
1315925663,5.760000000000,2.267000000000
1315927898,5.650000000000,2.542000000000
1315942379,5.920000000000,0.450000000000
1315942469,5.950000000000,3.578600000000
```

Slika 4.2: Odsek zgodovinskih podatkov.

Na sliki 4.2 je predstavljen odsek zgodovinskih podatkov oziroma transakcij za eno izmed borz v csv obliki. Prvi stolpec predstavlja čas transakcije v UNIX timestamp obliki. Drugi stolpec prestavlja ceno v USD, tretji pa količino bitcoinov. Ker bomo podprli več borz, smo za naše potrebe ustvarili še en stolpec z imenom borze, da bomo lahko poizvedovali po bazi glede na ime borze.

Te zgodovinske podatke smo nato uvozili v bazo s pomočjo ukaza *mongoimport* v ukazni vrstici. Vendar teh podatkov še ne moremo uporabiti za našo aplikacijo, saj jih potrebujemo v OHLC obliki, morajo pa biti tudi grupirani na različne časovne enote:

- 1 minuta,
- 1 ura,
- 1 dan,
- 1 teden,
- 1 mesec.

Tako je nastal program v Javi (OhlcHistoric.java), ki bere zgodovinske podatke enega za drugim iz baze in na podlagi izbrane časovne enote za vsako obdobje izračuna OHLC vrednosti ter jih shrani v bazo skupaj s časom obdobja v UNIX timestamp obliki. Podatke smo za vse borze začeli računati od 1. 10. 2011 naprej.

Za shranjevanje v bazo program pokliče metodo *SaveToMongoDB* razreda Helper. To je pomožni razred, ki je namenjen shranjevanju v bazo in je skupen vsem programom v Javi, ki smo jih razvili. Metoda *SaveToMongoDB* kot parametre dobi JSON objekt, ki ga je potrebno shraniti, ter ime zbirke, v katero se mora shraniti.

```
public static void SaveToMongoDB(JSONObject data, String
    collection_name) {
     try {
```

```
        //prvic naredimo povezavo z bazo
        if(connection==false){
          createConnection();
          connection=true;
        }
        //pridobimo zbirko, kamor bomo shranjevali
        DBCollection collection = getDB().getCollection(
            collection_name);
        // spremeni JSON v DBObject
        DBObject dbObject = (DBObject) JSON.parse(data.
            toString());
        //shranimo
        collection.insert(dbObject);
      } catch (Exception e) {
        logger.error(new SimpleDateFormat("yyyy/MM/dd HH:mm:
            ss").format(new Date())+" "+e);
        e.printStackTrace();
      }
    }
```

4.1.2 Transakcije

Poleg zgodovinskih trgovalnih podatkov potrebujemo še trenutne podatke oziroma transakcije in tako omogočimo, da bo naša aplikacija ves čas dobivala sveže podatke. Te pridobimo preko spletnega API-ja borz.

Do podatkov borze Bitstamp dostopamo s HTTP GET zahtevo `https://www.bitstamp.net/api/transactions/`, do podatkov borze Btc-e pa s HTTP GET zahtevo `https://btc-e.com/api/2/btc_usd/trades`. Ta zahteva vrne padajoči JSON seznam transakcij v zadnji uri. Vsaka transakcija vsebuje UNIX timestamp čas, ceno v USD ter količino.

Za shranjevanje teh podatkov smo napisali program v Javi (OhlcNew.java), ki vsakih nekaj sekund povprašuje po njih in sproti iz transakcij računa OHLC podatke za različne časovne enote, tako kot pri zgodovinskih podatkih, ter jih shranjuje v bazo. Program neprekinjeno teče na strežniku.

Spodaj je predstavljena koda, ki se poveže na API borze in prebere JSON seznam transakcij. Podobno dostopamo tudi do drugih podatkov, ki jih dobimo preko spletnega API-ja borz, potrebno je zamenjati samo URL naslov.

```
JSONArray data =getJSONfromURL("https://www.bitstamp.net/
    api/transactions/");
JSONArray data =getJSONfromURL("https://btc-e.com/api/2/
    btc_usd/trades");

public static JSONArray getJSONfromURL(String URL) throws
    NoSuchAlgorithmException, KeyManagementException{
        try{
            URLConnection uc;
            URL url = new URL(URL);
            uc = url.openConnection();
            uc.setConnectTimeout(1500);
            uc.addRequestProperty("User-Agent", "Mozilla
                /4.0 (compatible; MSIE 6.0; Windows NT 5.0)
                ");
            uc.connect();
            BufferedReader rd = new BufferedReader(new
                InputStreamReader(uc.getInputStream(),
                Charset.forName("UTF-8")));
            StringBuilder sb = new StringBuilder();
            int cp;
            while ((cp = rd.read()) != -1)
            {
                sb.append((char)cp);
            }
            String jsonText = (sb.toString());
            return new JSONArray(jsonText.toString());
        } catch (IOException ex){
          logger.warn(new SimpleDateFormat("yyyy/MM/dd HH:
              mm:ss").format(new Date())+" "+ex);
            return null;
        }
  }
```

4.1.3 Knjiga naročil

Podatke knjige naročil potrebujemo za graf globine trga posamezne borze ter izpis knjige naročil v spletni aplikaciji. Podatke pridobimo preko spletnega API-ja borz.

Do podatkov borze Bitstamp dostopamo s HTTP GET zahtevo `https://www.bitstamp.net/api/order_book/`, do podatkov borze Btc-e pa s HTTP GET zahtevo `https://btc-e.com/api/2/btc_usd/depth`. Ta zahteva vrne JSON slovar povpraševanja in ponudbe. Vsak je seznam odprtih naročil in vsako naročilo predstavlja seznam cene in količine.

Pridobljeni podatki so že pretežno v obliki, ki jo potrebujemo. Naš program (Orderbook.java) je narejen tako, da vsako minuto shrani v bazo nove podatke.

4.1.4 Ticker

Ticker nam vrne JSON slovar z naslednjimi podatki:

- last - zadnja BTC cena,
- high - najvišja cena zadnjih 24 ur,
- low - najnižja cena v zadnjih 24 urah,
- volume - promet v zadnjih 24 urah,
- bid - najvišje naročilo za nakup,
- ask - najnižje naročilo za prodajo.

Teh podatkov ne potrebujemo nujno. V naši aplikaciji smo jih uporabili za prikazovanje zadnje cene bitcoina na posamezni borzi. Sprva smo te podatke nameravali uporabiti namesto transakcij za računanje OHLC podatkov. Vendar se podatki osvežujejo enkrat na pol minute in vrnejo samo zadnjo vrednost, zato OHLC podatki, ki bi jih pridobili, ne bi bili tako natančni.

Do podatkov borze Bitstamp dostopamo s HTTP GET zahtevo `https://www.bitstamp.net/api/ticker/`, do podatkov borze Btc-e pa s HTTP GET zahtevo `https://btc-e.com/api/2/btc_usd/ticker`.

Tako kot pri knjigi naročil tudi program za shranjevanje tickerjev (Data.java) vsako minuto shrani v bazo nove podatke.

4.2 Podatkovna baza

Podatkovna baza MongoDB za naš informacijski sistem je sestavljena iz sedmih zbirk (glej sliko 4.3). Te med seboj niso povezane v smislu referenčnih integritet. Podatki so v zbirkah shranjeni v obliki JSON dokumentov. Spodaj si lahko ogledate strukturo teh zbirk na primeru enega dokumenta, za tem pa še celotno shemo podatkovne baze v obliki tabel. Pri tem je potrebno opozoriti, da podatki v NoSQL podatkovnih bazah, kot je tudi MongoDB, v resnici niso shranjeni v tabelah. Vseeno smo za lažjo predstavo naredili shemo podatkovne baze v obliki tabel, saj imajo vsi podatki, ki jih shranjujemo v posamezno zbirko, enako shemo.

- Zbirke OhlcMonth, OhlcWeek, OhlcDay, OhlcHour, OhlcMinute vsebujejo OHLC podatke, grupirane na časovno enoto posamezne zbirke.

```
{
    "_id" : ObjectId("5391208de4b063d374e80fd4"),
    "timestamp" : 1402019700,
    "open" : 659.99,
    "high" : 660,
    "low" : 659.99,
    "exchange" : "bitstamp",
    "close" : 660
}
```

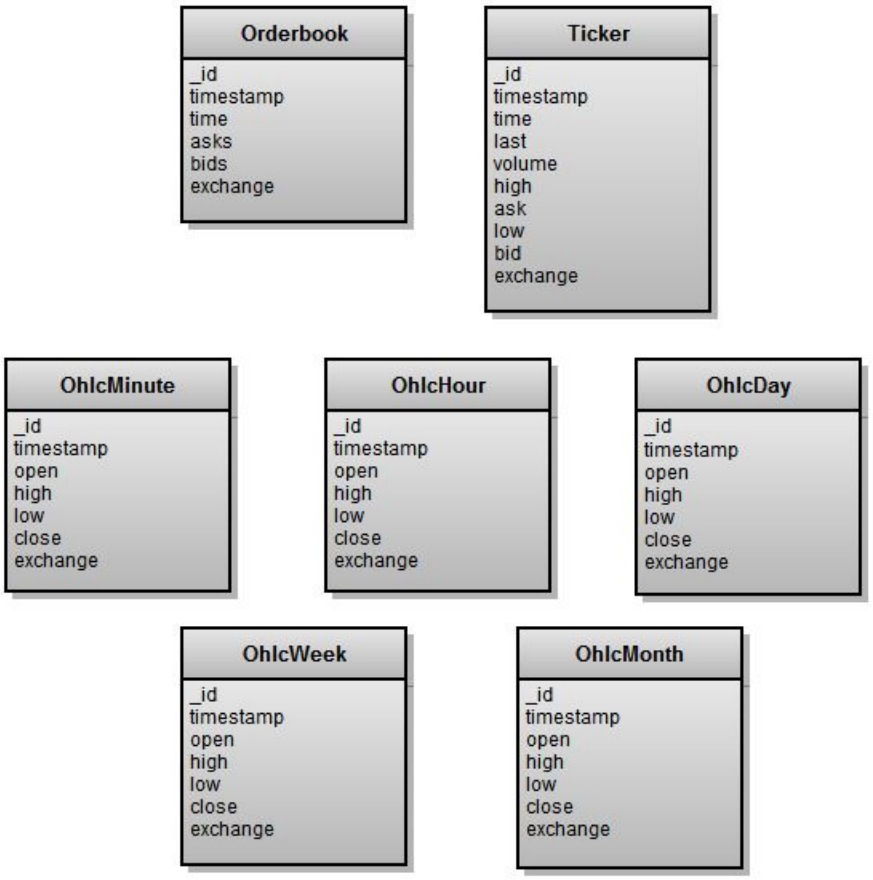
- Zbirka Orderbook vsebuje podatke o knjigi naročil, torej ponudbo in povpraševanje.

```
{
    "_id" : ObjectId("533c05c74a9207a94c06cc1d"),
    "timestamp" : 1396442567,
    "time" : 1396442520,
    "asks" : [
        [
            481.293,
            0.015
        ],
        ...
    ],
    "bids" : [
        [
            481.063,
            0.23
        ],
       ...
    ],
    "exchange" : "bitstamp"
}
```


- Zbirka Ticker vsebuje podatke o tickerjih.


```
{
    "_id" : ObjectId("533c044b4a92e824fea103b3"),
    "timestamp" : 1396442335,
    "time" : 1396442280,
    "last" : 476,
    "volume" : 14758.07203557,
    "high" : 494.98,
    "ask" : 479.56,
    "low" : 471.02,
```

```
    "bid" : 476,
    "exchange" : "bitstamp"
}
```



Slika 4.3: Shema podatkovne baze.

4.3 Spletna aplikacija

Ko smo končali s pridobivanjem podatkov in dobili ustrezno podatkovno bazo, smo začeli delati na aplikaciji v ogrodju Sails.js. Podrobnosti o samem

ogrodju in njegovi strukturi si lahko preberete v poglavju 3.1.6. Izvorna koda spletne aplikacije je na voljo na spletnem naslovu `https://bitbucket.org/stursic/bitcoincharts`.

Najprej je bilo potrebno povezati aplikacijo z našo podatkovno bazo. To smo storili tako, da smo v datoteko config/adapters.js ogrodja Sails vpisali naslednjo kodo:

```
module.exports.adapters = {
    'default': 'mongo',
      // sails v.0.9.0
      mongo: {
        module   : 'sails-mongo',
        host     : 'sandbox.lavbic.net',
        port     : 27017,
        user     : 'tursic',
        password : '****',
        database : 'dipl-tursic-btc'
      }
}
```

4.3.1 Vizualizacija pridobljenih podatkov

Za vizualizacijo podatkov smo uporabili knjižnico Highstock. Več o tej knjižnici si lahko preberete v poglavju 3.3.1.

Pridobljene podatke smo uporabili za dve vrsti vizualizacije:

- graf gibanja cene kriptovalute Bitcoin ter izris tehničnih indikatorjev nanj,
- graf globine trga in knjiga naročil.

Graf gibanja cene

Na sliki 4.4 imamo predstavljeno prvo vizualizacijo. Prikazuje graf gibanja cene kriptovalute Bitcoin in omogoča tudi povečanje in zmanjšanje območja

(angl. zoom in/out). Namesto cene so prikazani japonski svečniki. Uporabnik lahko izbira med več borzami. Na voljo so Bitstamp, Btc-e ter MtGox.

Slika 4.4: Graf gibanja cene bitcoina.

Japonski svečniki prikazujejo podatke o najvišji, najnižji, otvoritveni in zaključni vrednosti za določeno časovno obdobje. Zaradi tega smo morali podatke pridobiti v OHLC obliki. Telesa svečnikov so dveh barv. V našem primeru imajo telo modre barve svečniki, katerih zaključna vrednost je manjša od otvoritvene (medvedji svečniki), bele barve pa tisti, katerih zaključna vrednost je večja od otvoritvene (bikovski svečniki) [17]. Več o japonskih svečnikih si lahko preberete v poglavju 2.2.1.

Graf je implementiran z asinhronim nalaganjem podatkov. Podatkov je preveč, da bi lahko vse naenkrat naložili v graf, zato ga ob vsaki spremembi obdobja osvežimo z novimi podatki. Kot smo že omenili, smo grupirali OHLC podatke na različna časovna obdobja in jih shranili v bazo v ustrezno zbirko,

glede na grupirano obdobje. Na podlagi obdobja, ki ga uporabnik želi prikazati na grafu, se nato določi, iz katere zbirke bo črpal podatke, saj ne moremo na primer za obdobje enega leta prikazovati minutnih podatkov.

obdobje na grafu	zbirka	grupiranje podatkov
do 10 ur	OhlcMinute	na minuto
od 10 ur do 1 meseca	OhlcHour	na uro
od 1 meseca do 1.5 let	OhlcDay	na dan
od 1.5 let do 5 let	OhlcWeek	na teden
od 5 let naprej	OhlcMonth	na mesec

Tabela 4.1: Izbor grupiranja podatkov glede na obdobje na grafu.

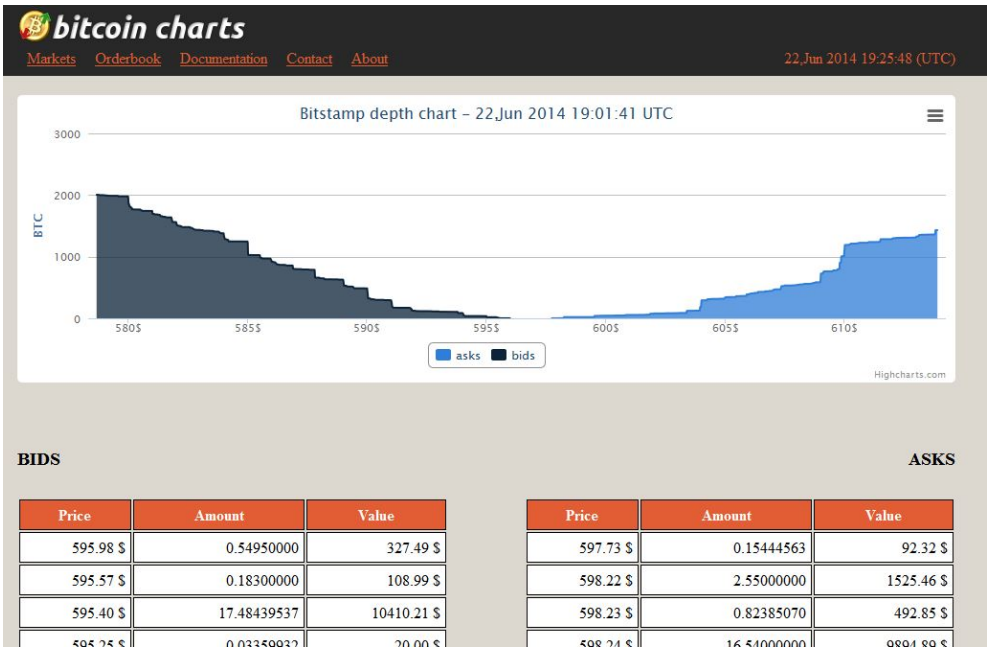
Ob vsaki spremembi obdobja se tako pokliče funkcijo *afterSetExtremes*, ki prebere nov začetek in konec obdobja in na podlagi teh novih ekstremov pridobi JSON seznam podatkov v tem obdobju. Podatke, ki so vrnjeni v spremenljivki *data*, nato uporabi graf.

```
function afterSetExtremes(e) {

  var currentExtremes = this.getExtremes(),
    range = e.max - e.min,
    chart = $('#chart').highcharts();

  chart.showLoading('Loading data from server...');

  $.getJSON('/bitcoinCharts/historicdata/databitstamp?
      start='+Math.round(e.min)+'&end='+Math.round(e.max),
      function(data) {

    chart.series[0].setData(data);
    chart.hideLoading();
  });

}
```

Na grafu je možno tudi vizualizirati tehnične indikatorje, ki smo jih implementirali s pomočjo Highstock vtičnikov. Uporabnik lahko izbira med indikatorji EMA (Exponential Moving Average), SMA (Simple Moving Average), MACD (Moving Average Convergence Divergence) ter RSI (Relative strength index). Možno je izbrati tudi več indikatorjev naenkrat. Na sliki 4.4 je prikazan MACD indikator.

Graf globine trga in knjiga naročil

Naša aplikacija omogoča tudi prikaz grafa globine trga ter izpis knjige naročil. To predstavlja slika 4.5. Podatki se osvežujejo vsako minuto.



Slika 4.5: Graf globine trga.

Graf globine trga nam omogoča grafični prikaz celotne ponudbe (angl. bids) in povpraševanja (angl. ask) na borzi. Ponudbe so nakupna naročila, povpraševanja pa prodajna naročila. Globina trga nam pove, kolikšna je še lahko velikost posla, ne da bi bila povzročena nenormalna tečajna gibanja. Manjša je globina trga, večja je možnost vplivanja na tečaj [40].

Z grafa zgoraj lahko kot primer razberemo, da lahko prodamo okoli 1000 bitcoinov dokler ne gre cena pod 585 $, kupimo pa lahko okoli 1000 bitcoinov dokler cena ne gre preko 610 $. Z drugimi besedami, za okoli 1000 bitcoinov imamo nakupnih naročil s ceno 585 $ in več in za okoli 1000 bitcoinov imamo prodajnih naročil s ceno 610 $ in manj.

Pod grafom pa je izpisana knjiga naročil, torej ponudbe (levo) in povpraševanja (desno). Pri ponudbah nam prvi stolpec pove, koliko so pripravljeni kupci plačati za en bitcoin v USD, drugi količino bitcoinov, ki jo želijo kupiti, in tretji vrednost transakcije. Ponudbe so sortirane od najvišje cene do najnižje. Pri povpraševanjih pa nam prvi stolpec pove, koliko denarja želijo imeti prodajalci za en bitcoin v USD, drugi količino bitcoinov, ki jo želijo prodati, in tretji vrednost transakcije. Povpraševanja so sortirana od najnižje cene do najvišje.

4.3.2 Izvajanje strategij

Ključna funkcionalnost naše aplikacije je možnost izvajanja simulacije trgovanja (angl. backtesting) na podlagi različnih strategij. Služi za testiranje strategij in je pomembna funkcija za trgovalce, saj jim pomaga spoznati, kako bi se posamezna strategija obnesla, in jim ponuja priložnost, da strategije izboljšajo [41]. Na sliki 4.6 je prikazana simulacija trgovanja na podlagi MACD strategije od 11. 7. 2013 do 8. 1. 2014.

Uporabnik mora najprej na grafu izbrati obdobje za katero želi izvesti simulacijo trgovanja. Začetni in končni datum se pridobita z grafa in se zapišeta v ustrezen prostor pod grafom. Nato mora samo še izbrati strategijo in vpisati količino denarja, s katero želi trgovati. Količina denarja je obvezen podatek, ki se validira. V primeru, da bo to okence prazno ali bo vanj vpisano kaj drugega kot število, sistem ne bo dovolil izvedbe simulacije. S pritiskom na gumb “Backtest” se izvede simulacija trgovanja, njeni rezultati pa se nato izpišejo, kot je prikazano na sliki 4.6. V našem primeru smo vložili 1000 $, na koncu simulacije po številnih nakupih in prodajah, ki jih je svetovala naša strategija, pa imamo na računu 14.39 BTC oziroma 11851.41 $. Poleg

Slika 4.6: Simulacija trgovanja z MACD strategijo.

izpisa rezultatov se na grafu vizualizirajo trgovalni signali. Zelena zastavica pomeni kupi, rdeča pa prodaj.

Pri implementaciji strategij smo si pomagali s knjižnico Talib, ki vsebuje funkcije za računanje različnih tehničnih indikatorjev in vzorcev na podlagi japonskih svečnikov. Vsaka strategija je shranjena v svoji datoteki in vsebuje nakupni signal, prodajni signal ter parametre Talib funkcij, ki jih uporablja. Trenutno so implementirane naslednje strategije:

Brez strategije (angl. No strategy): Kot že ime pove, ne gre za pravo strategijo. Simulacija kupi bitcoine na samem začetku izbranega obdobja in jih proda na koncu obdobja.

RSI strategija: RSI strategija temelji na RSI indikatorju. Podrobnosti o tem indikatorju in njegovih prodajnih in nakupnih signalih si lahko preberete v poglavju 2.2.2. Pri implementaciji te strategije smo spremenili nekatere parametre, kot so običajno. Mejo preprodanosti smo postavili na 40, mejo prekupljenosti pa na 60, saj tako strategija daje boljše rezultate.

MACD strategija: Ta strategija temelji na MACD tehničnem indikatorju, ki kaže spremembe v trendu. Tudi o tem indikatorju in njegovih nakupnih ter prodajnih signalih si lahko podrobnosti preberete v poglavju 2.2.2.

CDLHARAMI strategija: Poznamo kar nekaj vzorcev japonskih svečnikov, eden izmed njih je tudi harami, na katerem temelji ta strategija. Več informacij o japonskih svečnikih in harami vzorcu dobite v poglavju 2.2.1.

Strategija po meri (angl. Custom strategy): Uporabnik ima možnost napisati tudi svojo strategijo in jo uporabiti za simulacijo trgovanja. Ob izbiri te strategije, se le ta prebere iz datoteke in prikaže v polju za vnos, kjer jo lahko nato uporabnik spreminja in tudi shrani. Dokumentacijo o funkcijah iz knjižnice Talib, ki jih lahko uporabite, in njenih parametrih najdete na povezavi “Documentation” v meniju naše aplikacije.

Specifikacije pri pisanju strategije:

- uporabite lahko do dve funkciji,
- ni potrebno pisati parametrov open, high, low, close, endIdx ter inReal,
- trenutni indeks: i,
- začetni indeks: startindex,
- končni indeks: array.timestamp.length-1,
- dostop do izhodnega parametra: result.result.outputParameter[i] za params ter result1.result.outputParameter[i] za params1, imena izhodnih parametrov so v dokumentaciji za vsako funkcijo,
- dostop do trenutne vrednosti bitcoina: array.close[i],
- dostop do trenutnega timestamp časovnega podatka: array.timestamp[i].

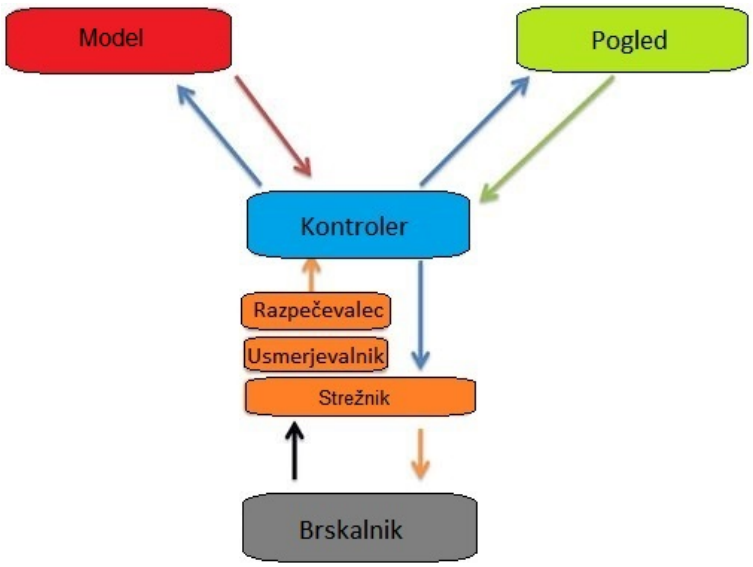
```
//primer: MACD strategija
var params={
  name: "MACD",
  startIdx: 0,
  optInFastPeriod: 12,
  optInSlowPeriod: 26,
  optInSignalPeriod: 9 };
var params1={};
//prodajni signal: ko MACD pade pod MACD signalno crto
var sell="result.result.outMACD[i+1]< result.result.
   outMACDSignal[i+1]
     && result.result.outMACD[i]> result.result.
        outMACDSignal[i]";
//nakupni signal: ko se MACD dvigne nad MACD signalno crto
var buy="result.result.outMACD[i+1]>result.result.
   outMACDSignal[i+1]
    && result.result.outMACD[i]< result.result.
       outMACDSignal[i]";
```

4.3.3 Arhitekturni vzorec MVC

Aplikacijo smo razvili na Sails.js ogrodju, ki temelji na MVC arhitekturnem vzorcu [42]. Ta skrbi za ločitev poslovne logike od prezentativne plasti, katere namen je prikaz informacij uporabniku. Koda je (na ta način) organizirana tako, da skrije kompleksnost in omogoča lažje vzdrževanje aplikacije. MVC arhitektura je sestavljena iz treh delov:

- model - skrbi za interakcijo s podatkovno bazo,
- pogled - zadolžen za prikaz podatkov uporabniku,
- kontroler - logika, ki skrbi za komunikacijo med modelom in pogledom.

Implementacije MVC arhitekture se lahko razlikujejo glede na uporabljeno ogrodje ali programski jezik. Na sliki 4.7 je prikazana shema MVC arhitekture ogrodja Sails.js.



Slika 4.7: Shema MVC arhitekture.

Model je prezentacija podatkov, shranjenih v podatkovni bazi. Njegova naloga je, da se pogovarja s podatkovno bazo, torej shranjuje, poizveduje, validira podatke. V našem primeru je model definiran kot JavaScript razred s pripadajočimi atributi. Vsaka zbirka v bazi se tako preslika v svoj razred.

Kontroler v Sails.js deluje podobno kot v ostalih MVC ogrodjih. Je vmesni člen med modelom in pogledom. Zgrajen je kot JavaScript razred, ki vsebuje metode (akcije), ki se pokličejo, ko brskalnik poda zahtevo za prikaz neke spletne strani.

Pogled je namenjen prikazu podatkov. Ponavadi gre za html datoteke, pri Sails.js pa so to datoteke s končnico ejs (embedded javascript).

MVC arhitektura deluje na naslednji način. Na začetku brskalnik pošlje zahtevo strežniku za prikaz spletne strani. Strežnik sprejme zahtevo in jo prek razpečevalca pošlje ustreznemu kontrolerju, da izvede določeno akcijo. Ta akcija je tehnično gledano metoda kontrolerja. Ime kontrolerja in akcije nam pove usmerjevalnik (datoteka config/routes.js), ki spletne naslove oziroma poti pretvori v pripadajoči kontroler in akcijo. Poklicana metoda kontrolerja nato zahtevo obdela, če je potrebno, zahteva od modela določene podatke, ki jih nato vrne kontrolerju. Ta nato poskrbi, da se podatke pošlje naprej do pogleda, ki je zadolžen za prikaz spletne strani. Kontroler na koncu pošlje strežniku metapodatke in to, kar je generiral pogled. Strežnik vse te podatke primerno zapakira in pošlje brskalniku kot odgovor na njegovo zahtevo [43].

Poglavje 5

Sklepne ugotovitve

V diplomskem delu smo razvili prototip informacijske rešitve, ki omogoča izvajanje različnih trgovalnih strategij nad kriptovaluto Bitcoin. Rešitev je bila razvita v tehnologiji Node.js kot spletna aplikacija. Odločili smo se podpreti največje borze: Bitstamp, Btc-e in MtGox.

Na začetku razvoja smo ugotovili, da so potrebni podatki o trgovanju podprtih borz javno dostopni na spletu. Večino potrebnih podatkov (knjiga naročil, transakcije, ticker) pridobivamo preko spletnih API-jev podprtih borz s pomočjo programov, napisanih v Javi in jih shranjujemo v podatkovno bazo MongoDB, zgodovinske podatke pa smo dobili v obliki csv datotek na spletnem mestu BitcoinCharts.

Pridobljene podatke smo uporabili pri razvoju spletne aplikacije, ki temelji na arhitekturi MVC. Za vsako podprto borzo smo realizirali graf, ki prikazuje vrednost bitcoina skozi čas v obliki japonskih svečnikov. Uporabnik lahko na tem grafu vizualizira različne tehnične indikatorje. Poleg tega aplikacija omogoča tudi prikaz globine trga in knjige naročil. Za vizualizacijo je bila uporabljena knjižnica Highstock, ki je ena izmed vodilnih na tem področju. Pojavilo se je tudi vprašanje, kako implementirati strategije. Tu smo si pomagali s knjižnico Talib, ki vsebuje funkcije za računanje tehničnih indikatorjev. Aplikacija že ima implementiranih nekaj preddefiniranih strategij, uporabnik pa ima možnost napisati in uporabiti tudi svojo strategijo. Ob

izvedbi simulacije se poleg izpisanih rezultatov generirani trgovalni signali tudi vizualizirajo na grafu.

Cilje, ki smo si jih zastavili, smo tako dosegli. Nastala rešitev je javno dostopna na spletnem naslovu `sandbox.lavbic.net/bitcoinCharts/`. Je odprtokodna, brezplačna in je lahko v veliko pomoč tistim, ki se ukvarjajo s trgovanjem s to kriptovaluto. Med razvojem informacijske rešitve sem bolj podrobno spoznal kriptovaluto Bitcoin, trgovanje pa tudi nekatere tehnologije, ki sem jih pri izdelavi diplomskega dela prvič uporabil (Node.js, MongoDB), kar je bil tudi eden izmed namenov tega diplomskega dela.

5.1 Mogoče izboljšave

Informacijska rešitev, ki smo jo izdelali, je prototip in predstavlja osnovo, na kateri se lahko gradi naprej. Odprtih je še kar nekaj možnosti za izboljšave:

- Podpora več borzam in tudi drugim kriptovalutam.
- Izboljšanje dodajanja in pisanja novih strategij. V trenutni verziji imamo implementirano strategijo po meri, ki jo lahko vsak spreminja, kakor želi. Pri tem se pojavlja tudi problem varnosti. Dodala bi se možnost registracije in prijave uporabnika v sistem. Vsak registrirani uporabnik bi lahko dodajal svoje strategije, jih pregledoval v posebnem pogledu in jih tudi delil z ostalimi uporabniki. Tisti, ki se ne bi želeli registrirati, ne bi imeli možnosti dodajanja svojih strategij, ampak bi lahko uporabili le tiste, ki so že preddefinirane v aplikaciji.
- Prikaz najboljših strategij po izboru uporabnikov.
- Možnost komentiranja strategij.
- Trenutno je mogoča samo simulacija izvajanja strategij na zgodovinskih podatkih (angl. backtesting). Lahko bi dodali še možnost predikcije, sistem bi izvajal izbrano strategijo in preko elektronske pošte sporočal trgovalne signale, torej kdaj je čas za prodajo oziroma nakup.

- Videz same aplikacije ni bil prioriteta pri razvoju naše informacijske rešitve. Zaradi tega bi se lahko lotili tudi prenove grafičnega vmesnika, saj je danes tudi to zelo pomemben vidik spletne aplikacije.

Literatura

[1] Maja Žiberna. (2014) Je bitcoin elektronski denar prihodnosti? Dostopno na:
http://www.rtvslo.si/znanost-in-tehnologija/je-bitcoin-elektronski-denar-prihodnosti/337900

[2] Fakulteta za matematiko in fiziko. (2014) Bitcoin in druge kriptovalute. Dostopno na:
http://www.fmf.uni-lj.si/storage/29638/Bitcoinindrugekriptovalute-vsebinaprojekta.pdf?access=9d20adb719970d7ceff9426cc1e1da9d

[3] (2014) Trgovalna platforma Plus500. Dostopno na:
www.plus500.si/

[4] (2014) UpBit: Cryptocurrency. Dostopno na:
https://upbit.org/en/what-is-it/

[5] S. Nakamoto. (2008) Bitcoin: A Peer-to-Peer Electronic Cash System. Dostopno na:
http://nakamotoinstitute.org/bitcoin/

[6] Nathan Willis. (2010) Bitcoin: Virtual money created by CPU cycles. Dostopno na:
http://lwn.net/Articles/414452/

[7] (2014) Bitcoin faq. Dostopno na:
https://bitcoin.org/en/faq

[8] (2014) Bitcoin wiki. Dostopno na:
https://en.bitcoin.it/wiki/Main_Page

[9] (2014) Borza Bitstamp. Dostopno na:
https://si.bitstamp.net/

[10] (2014) Kako bitcoin deluje? Dostopno na:
https://bitcoin.org/sl/kako-deluje

[11] Chukwumah Ezeobika. (2013) Advantages and Disadvantages of Using Bitcoins. Dostopno na:
http://web.archive.org/web/20131201164752/http://voices.yahoo.com/advantages-disadvantages-bitcoin-12385858.html?cat=15

[12] F.Reid and M. Harrigan,"An analysis of anonymity in the bitcoin system", v *Security and Privacy in Social Networks*, New York: Springer, 2013, str. 197-223.

[13] C. Decker and R. Wattenhofer. (2013) Bitcoin Transaction Malleability and MtGox. Dostopno na:
http://arxiv.org/pdf/1403.6676.pdf

[14] Steven B. Achelis, *Technical Analysis from A to Z*, New York: McGraw Hill, 2001.

[15] (2014) Investopedia: Technical analysis. Dostopno na:
http://www.investopedia.com/university/technical/

[16] (2014) StockCharts: Technical analysis. Dostopno na:
http://stockcharts.com/school/doku.php?id=chart_school:overview:technical_analysis

[17] (2014) Candlestick chart. Wikipedia. Dostopno na:
http://en.wikipedia.org/wiki/Candlestick_chart

[18] (2014) Online Trading Concepts: Candlestick patterns. Dostopno na: http://www.onlinetradingconcepts.com/TechnicalAnalysis/Candlesticks/CandlestickBasics.html

[19] G. Caginalp and H. Laurent, "The predictive power of price patterns", *Applied Mathematical Finance*, št. 3, zv. 5 , str. 181-205, 1998.

[20] (2014) Finančni trgi: Kateri indikator izbrati pri trgovanju? Dostopno na: http://www.financnitrgi.com/trgovanje/trgovalna-knjiznjica/kateri-indikator-izbrati-pri-trgovanju-magicnih-namrec-ni

[21] (2014) Investopedia: Moving average. Dostopno na: http://www.investopedia.com/terms/m/movingaverage.asp

[22] (2014) TradingSim: Exponential moving average. Dostopno na: http://tradingsim.com/blog/exponential-moving-average-ema-technical-indicator/

[23] (2014) Online Trading Concepts: Technical indicators. Dostopno na: http://www.onlinetradingconcepts.com/TechnicalAnalysis.html

[24] (2014) Investopedia: Relative Strength Index - RSI. Dostopno na: http://www.investopedia.com/terms/r/rsi.asp

[25] (2014) Gekko. Dostopno na: https://github.com/askmike/gekko

[26] (2014) Cryptotrader. Dostopno na: https://cryptotrader.org/

[27] (2014) Java (programming language). Wikipedia. Dostopno na: http://en.wikipedia.org/wiki/Java_programming_language

[28] (2014) Oracle: The Java Language Environment. Dostopno na: http://www.oracle.com/technetwork/java/index-136113.html

[29] Zoran Bosnić, gradivo predmeta "Spletno programiranje", 2012.

[30] (2014) JavaScript. Wikipedia. Dostopno na:
http://sl.wikipedia.org/wiki/JavaScript

[31] (2014) Nodejs.org. Dostopno na:
http://nodejs.org/

[32] Pedro Teixeira, *Professional Node.js: Building Javascript Based Scalable Software*, John Wiley & Sons, 2012.

[33] (2014) Sails.js. Dostopno na:
http://sailsjs.org/

[34] (2014) MongoDB. Dostopno na:
http://www.mongodb.org/

[35] Neal Leavitt, "Will NoSQL Databases Live Up to Their Promise?", *Computer*, št. 2, zv. 43, str. 12-14, 2010.

[36] (2014) Eclipse (software). Wikipedia. Dostopno na:
http://en.wikipedia.org/wiki/Eclipse_(software)

[37] (2014) Robomongo. Dostopno na:
http://robomongo.org/

[38] (2014) Highcharts. Dostopno na:
http://www.highcharts.com/

[39] (2014) Node-talib. Dostopno na:
https://github.com/oransel/node-talib

[40] (2014) Cekin: Globina trga. Dostopno na:
http://cekin.si/clanek/slovar/globina-trga.html

[41] J. Ni and C. Zhang,"An Efficient Implementation of the Backtesting of Trading Strategies", v *Parallel and Distributed Processing and Applications*, Springer Berlin Heidelberg, 2005, str. 126-131.

[42] (2014) Tutorialspoint: Basic MVC architecture. Dostopno na: http://www.tutorialspoint.com/struts_2/basic_mvc_architecture.htm

[43] Kalid Azad. (2007) Intermediate Rails: Understanding Models, Views and Controllers. Dostopno na: http://betterexplained.com/articles/intermediate-rails-understanding-models-views-and-controllers/